

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
10 April 2003 (10.04.2003)

PCT

(10) International Publication Number
WO 03/029996 A1

(51) International Patent Classification⁷: **G06F 13/24**,
13/42

(72) Inventors: **CHANG, Yeow, K.**; Prof . Holstlaan 6,
NL-5656 AA Eindhoven (NL). **ZHANG, Zhenyu**; Prof .
Holstlaan 6, NL-5656 AA Eindhoven (NL).

(21) International Application Number: PCT/IB02/03812

(74) Agent: **DUIJVESTIJN, Adrianus, J.**; Internationaal Oc-
trooibureau B.V., Prof. Holstlaan 6, NL-5656 AA Eind-
hoven (NL).

(22) International Filing Date:
16 September 2002 (16.09.2002)

(25) Filing Language: English

(81) Designated States (*national*): CN, JP, KR.

(26) Publication Language: English

(84) Designated States (*regional*): European patent (AT, BE,
BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT,
LU, MC, NL, PT, SE, SK, TR).

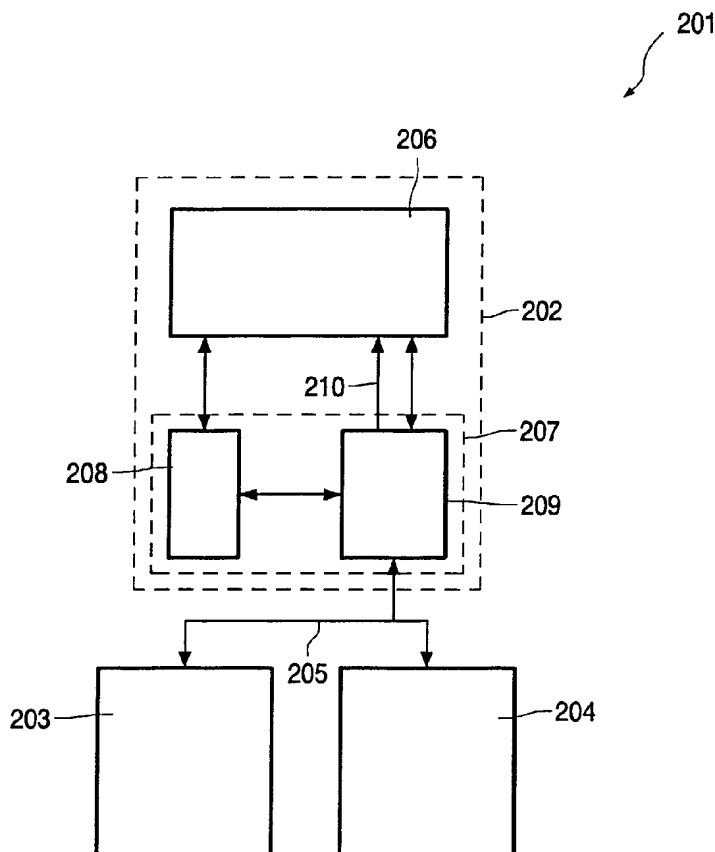
(30) Priority Data:
PCT/SG01/00194
27 September 2001 (27.09.2001) SG

Published:
— with international search report

(71) Applicant: **KONINKLIJKE PHILIPS ELECTRON-
ICS N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA
Eindhoven (NL).

For two-letter codes and other abbreviations, refer to the "Guid-
ance Notes on Codes and Abbreviations" appearing at the begin-
ning of each regular issue of the PCT Gazette.

(54) Title: BUS SYSTEM AND BUS INTERFACE FOR CONNECTION TO A BUS



(57) Abstract: The invention relates to a bus system comprising a first station (202) and a second station (203, 204) coupled by a bus for transferring data and control signals. The bus is conceived to operate according to a protocol in which the first station (202) repeatedly sends requests to the second station (203, 204). The first station (202) comprises a processor (206), a bus interface (207), and a buffer (208) coupled to said interruptible processor and said bus interface, the processor (206) being operable to generate request properties for the requests to be sent by the first station (202). Furthermore, the processor (206) is operable to handle said data items. The buffer (208) is operable to store said request properties. The bus interface (207) is operable to generate said requests from said stored request properties and to repeatedly send said requests.



WO 03/029996 A1

Bus system and bus interface for connection to a bus

The invention relates to a bus system comprising a first station and a second station coupled by a bus for transferring data and control signals, said bus being conceived to operate according to a protocol in which said first station repeatedly sends requests to said second station.

5 The invention also relates to a bus interface, comprising a connection for a bus and an input for receiving request properties from a processor.

Such a bus system is disclosed in the USB specification. The USB
10 specification is a standard promulgated by a group of companies including Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, and Koninklijke Philips Electronics N.V. Described below are various aspects of the USB relevant to a complete understanding of the present invention. Further background concerning the USB may be obtained from USB
15 Specification, Revision 2.0, which is included herein by reference.

The USB is a standard bus designed to support data exchange between a personal computer and a wide variety of peripherals such as webcams, printers, scanners, microphones and keyboards. The USB thus replaces existing interfaces such as the RS-232C serial ports, parallel ports, and PS/2 interfaces. Further developments of the USB, for
20 example, as in USB On-The-Go, aim at bus systems where another apparatus replaces the personal computer. Such an apparatus could be, for example, a digital still camera, a video camera, a personal digital assistant, or a cellular phone.

In a bus system like a USB system, electronic bus stations are connected to one another via a bus. Usually a bus station comprises a processor or computing system to
25 process or generate data and to control the flow of data. USB stations are defined as hosts (e.g. personal computers) and as devices (peripherals). In general there is only one host on any USB system. A USB interface at the host is also referred to as a host controller. The host controller may be implemented in a combination of hardware, firmware, or software. USB devices are defined as hubs which provide additional attachment points or functions which

provide capabilities to the system. The devices are connected to the host via a tiered-star topology. Hubs indicate the attachment or removal of a USB device. The host determines if a newly attached USB device is a hub or a function and assigns a unique USB address to the USB device. All USB devices are accessed by a unique USB address. Each device
5 additionally supports one or more endpoints with which the host may communicate. The following description will be limited to USB devices defined as functions.

The USB is a polled bus. The host controller initiates all data transfers. Most bus transactions involve the transmission of up to three packets. Each transaction begins when the host controller, on a scheduled basis, sends a USB packet describing the type and
10 direction of the transaction, the USB device address, and endpoint number. This packet is referred to as the "token packet". The USB device that is addressed selects itself by decoding the appropriate address fields. In a given transaction, data is transferred either from the host to a device or from a device to the host. The direction of data transfer is specified in the token packet. The source of the transaction then sends a data packet or indicates it has no data to
15 transfer. The destination, in general, responds with a handshake packet indicating whether the transfer was successful.

The USB architecture comprehends three basic types of data transfer: (1) isochronous transfers: periodic, continuous communication between host and device, typically used for time-relevant information, which occupies a pre-negotiated amount of USB
20 bandwidth with a pre-negotiated latency; (2) interrupt transfers: low-frequency, bounded latency communication, typically used for asynchronous data such as characters or coordinates with human perceptible echo or feedback response characteristics, and; (3) bulk transfers: non-periodic, large-packet bursty communication, typically used for data that can use any available bandwidth and can also be delayed until bandwidth is available.

Usually both types of bus station, host and device, comprise a processor or computing system to either process or generate data or to control the flow of data or both. A straightforward implementation of a USB system as described above could lead to a significant overhead for both the processor in the bus station acting as a host and the processor in the bus station acting as a device. This holds especially for the interrupt
30 transfers. Due to the polled nature of the bus system, the host needs to interrogate the involved device, for instance a keyboard, on a regular basis.

It is inter alia an object of the invention to reduce the amount of overhead needed to transport data from a processor in a first station of a bus system to a second station of the bus system or vice versa.

To this end the invention provides a bus system as defined in the opening paragraph which is characterized in that said first station comprises a processor, a bus

5 interface, and a buffer coupled to said interruptible processor and said bus interface, where

- said processor is operable to generate request properties for said requests and for handling said data items;

- said buffer is operable to store said request properties;

10 - said bus interface is operable to generate said requests from said stored request properties and to repeatedly send said requests.

In operation the processor within the first station needs to provide information to the second station or needs to obtain information from the second station. To this end, the first station needs to communicate with the second station. In a polled bus system, such as a

15 USB system, the communication is initiated by the first station, being a host in a USB system. If the second station, being a device in a USB system, is only capable of receiving or providing data on an irregular basis, it happens frequently that the first station needs to poll the second station quite a number of times before a successful transfer of information takes place. In other words under these circumstances the first station needs to re-send the same

20 request relatively often to the second station before a successful transfer of information takes place.

In a conventional approach said processor would need to prepare said request every time it needs to be send and said bus interface would then actually send said request via said bus to said second station. In the bus system according to the invention the processor

25 generates or determines the properties of the request. By enhancing the capabilities of the bus interface, said bus interface can actually generate the request out of said properties provided by the processor and send the request over the bus to said second station as many times as is required.

The bus system according to the invention also comprises a buffer within the

30 first station that is used to store the request properties provided by the processor. The buffer is preferably accessible for both the processor and the bus interface. In this way the communication overhead of the processor is reduced.

An embodiment of the bus system according to the invention is characterized in that said request properties comprise a polling rate of said requests and, if applicable, a

payload of said requests. Including the polling rate in the request properties allows more flexibility. The processor may have information on the characteristics of the second station that can be used to adjust the polling rate of the bus interface. For instance, a USB system establishes a time base of either 1 millisecond, called a frame, or of 125 microsecond, called a microframe. If the polling rate information were not included in the request properties, the bus interface would use a fixed polling rate, presumably sending a request at the maximum polling rate of once per (micro)frame. The polling rate information being included in the request properties, the bus interface adjusts the polling rate as needed, using low polling rates whenever possible and high polling rates when required. Bus load and communication overhead for the second station are thus reduced.

In case the first station sends a request to receive data to the second station the request itself does not necessarily contain a payload, i.e. a data item. Including payload information in the request properties that are provided by the processor has the advantage that data can be sent repeatedly from the first station to the second station, without requiring interference from the processor within the first station, thus further reducing the communication overhead of the processor.

A further embodiment of the bus system according to the invention is characterized in that said second station responds to said requests, that said processor is an interruptible processor and said bus interface is operable for interrupting said processor upon reception of selected responses of said second station. It is not necessary to interrupt said processor if no further action is required. Thus it is not necessary to report back the result of the communication with the second station if either the communication is completed as expected or if no new information, no data items, are provided by the second station. For example, in USB system the second station may respond by sending a message with a data item acknowledge signal, a negative acknowledge signal, by sending a signal to indicate it is stalled. Furthermore, if the second station does not respond by sending a signal, or if no signal is received by the first station due to an error in the communication between the first station and the second station, this may be interpreted as a response too. Thus the selected responses may comprise the absence of a response signal from the second station. Communication overhead of the processor is further reduced if the processor is only interrupted upon reception of the responses that require further action by the processor. To this end the bus interface selects those responses out of the received responses. Programmed to handle such situations the processor can decide upon necessary further actions.

A bus interface according to the invention comprises a connection for a bus and an input for receiving request properties from an interruptible processor and is characterized in that it comprises a buffer operable to store said request properties and a controller operable to generate requests out of the stored request properties and to repeatedly
5 send said requests to the connection.

These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

10 Fig. 1 is a diagram showing the interrupt transaction format as specified in the USB specification, revision 2.0;

Fig. 2 is a block diagram showing a bus system according to the invention;

Fig. 3A is a message diagram showing transactions for data transmission;

Fig. 3B is a message diagram showing transactions for data transmission;

15 Fig. 4 is a block diagram showing a bus interface according to the invention;
and

Fig. 5 is schematic overview of a buffer organized in a first-in, first-out manner.

In these figures identical parts are denoted by identical references.

20 Fig. 1 is a diagram showing the interrupt transaction format as specified in the USB specification, revision 2.0. Within USB all communication is initiated by the host. All USB devices are accessed by a unique USB address. Each device additionally supports one or more endpoints with which the host may communicate.

The USB supports functional data and control exchange between the USB host and USB devices. USB data transfers take place between host software and a particular endpoint on a USB device. The USB architecture comprehends three basic types of transfer:

- isochronous transfers: isochronous or streaming real time data which occupies
30 a prenegotiated amount of USB bandwidth with a pre-negotiated latency;
- interrupt transfers: asynchronous interactive data such as characters or coordinates with human perceptible echo or feedback response characteristics, and

bulk transfers: asynchronous block transfer data which is generated or consumed in relatively large and bursty quantities and has wide dynamic latitude and transmission constraints.

If there are no on-going transactions, a USB system is in an Idle mode 101.

- 5 Bus transactions including data transfers generally involve three stages: a Setup stage 102, a Data stage 103, and a Status stage 104.

Bus transactions including data transfers generally involve the transmission of three different packets, e.g. a Token packet 105, 106, a Data packet 107, 108, and a Handshake packet 109, 110, 111, 112, 115, 116.

- 10 Interrupt transactions may consist of IN transfers or OUT transfers. In the case of an IN transfer the USB host will start the transaction by sending an IN token 105. Upon receipt of the IN token, a USB device may return data 107, a negative acknowledge signal (NAK) 111, or a signal indicating the USB device is unable to complete a command (STALL) 112. If the endpoint has no new interrupt information to return (i.e. no interrupt is pending), the USB device returns a NAK handshake 111 during the Data stage 103. If the
15 USB device is unable to complete a command, the USB device returns a STALL handshake 112. If an interrupt is pending, the USB device returns the interrupt information as a Data packet 107. The USB host in response to receipt of the Data packet 107 issues either an ACK handshake 109 (acknowledge signal) if the data was received error-free or returns no
20 handshake, indicated by line 113, if the Data packet 107 was received corrupted. Furthermore the USB host will not issue an ACK 109 in case no response to the IN token 105 is received at all from the USB device. This situation is indicated by line 114. After completion of the IN transfer, the USB system returns to the Idle mode 101.

- In the case of an OUT transfer the USB host will start the transaction by
25 sending an OUT token 106. This is followed by sending a Data packet 108. Upon receipt of the Data packet 108, the USB device may issue in the Status stage 104 an ACK handshake 110 if the data was received error free, a NAK handshake 115 indicating the device is not able to process the data, or a STALL handshake 116 indicating the USB device is currently stalled. The USB device returns no handshake, indicated by line 117, if the Data packet 107
30 was received corrupted. Furthermore the USB device will not issue an ACK handshake 110 in case it does not receive data from the USB host, a situation indicated by line 118. After completion of the OUT transfer the USB system returns to the Idle mode 101.

All packets begin with a SYNC field, which is used by input circuitry to align incoming data with a local clock. A packet identifier (PID) immediately follows the SYNC

field of every USB packet. The PID consists of a four-bit ID field followed by a four-bit check field. The PID indicates the type of packet (e.g. Token 105, 106, Data 107, 108, and Handshake 109, 110, 111, 112, 115, 116) and the format of the packet along with the type of variant detection applied to the packet. The four-bit check field of the PID ensures reliable decoding.

A Token packet 105, 106 includes an ADDR field which specifies the target USB device, via its USB address, that is the source or destination of a Data packet 107, 108, depending on the value of the Token PID. USB devices must perform a complete decoding of the ADDR field.

A Data packet 107, 108 includes a data field which may range from zero to N bytes and must amount to integral numbers of bytes. Data bits within each byte are shifted out most significant bit first. The data field of a Data packet is divided into fields which give extra information to the USB device regarding the request being made. One field, the command code, tells what the targeted USB device should do, such as set up a read from a USB device space or set up a write to a USB device space. The read space command is used to request information from a given location in a given space of a USB device. The command is used to set up the read, which is followed by a request from the host for the USB device to send data. The USB device then sends the data it has retrieved from the previously specified space. The write space uses exactly the same definition as the read space. However, the Data packet contains data transferred by the host to the USB device.

A Handshake packet 109, 110, 111, 112, 115, 116 consists of only a PID. As indicated above, Handshake packets are used to report the status of a data transfer and can return values indicating successful reception of data, CRC failure, flow control, and certain fault conditions.

Fig. 2 is a block diagram showing a bus system according to the invention. The bus system 201 comprises a first station 202, acting as a USB host, and second stations 203 and 204, acting as USB devices. The stations 202, 203, 204 are coupled by bus connections 205. The first station 202 is shown in more detail. It comprises an interruptible processor 206 and a bus interface 207. The bus interface comprises a buffer 208 and a controller 209. It is not necessary that the buffer is part of the bus interface. In an alternative arrangement the buffer could be placed within the first station 202, but outside the bus interface. Both the processor 206 and the controller 209 have access to the buffer 208. The controller is coupled to the bus connections 205. Furthermore the controller 209 has an interrupt connection 210 to the processor 206.

Fig. 3A is a message diagram showing transactions for data transmission. The message diagram illustrates the operation of the system of Fig. 1 for interrupt transactions consisting of IN transfers. Going top-down the time increases. Information exchange between the processor 206 and the bus interface 207 is symbolized by arrows between a first line 301 and a second line 302. Information exchange between the bus interface and the second station 203, 204 is symbolized by arrows between the second line 302 and a third line 303.

The processor 206 requires data from one of the second stations 203, 204, a USB device. Thus the bus interface 207 should repeatedly issue a request for data to the bus connections 205. To reduce the communication overhead for the processor it should be sufficient to have the processor provide the request properties only once. If the addressed USB device returns data, the bus interface 207 should interrupt the processor via the interrupt connection and provide the processor with the data. In case an error occurs the processor should also be notified. In these situations the bus interface 207 should apply an interrupt to the processor too. In other situations no action is required from the processor, so that the bus interface 207 should not interrupt the processor.

In an IN transfer 304 the processor 206 in a message 309 sends the request properties to the bus interface 207, where the request properties are stored in the buffer. Based on the request properties within the message 309, which are retrieved from the buffer, the bus interface 207 initiates the IN transfer by issuing an IN token 310 on the bus. The second station, the addressed USB device, responds by sending a message 311 with a data item. Upon reception of this message the bus interface 207 sends an interrupt signal 312 via the interrupt connection 210, to the processor 206 to inform the processor 206 that new data is available. Furthermore, it issues an Acknowledge signal 313 to the addressed USB device. This completes the IN transfer 304.

In an IN transfer 305, following the IN transfer 304 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer, while no new information is required from the processor 206. The bus interface 207 initiates the IN transfer by issuing an IN token 314 on the bus. The addressed USB device responds by sending a message 315 with a data item. Due to an error on the bus, the second message 315 is received corrupted by the bus interface 207. Therefore, the bus interface 207 does not issue an ACK handshake. Instead it sends an interrupt signal 316, via the interrupt connection 210, to the processor 206 to inform the processor 206 an error occurred. This completes the IN transfer 305.

In an IN transfer 306, following the IN transfer 304 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the IN transfer by issuing an IN token 317 on the bus. Due to a bus error the IN token 317 is received corrupted at the addressed USB device. For this reason it does not respond to the IN token 317. As no response follows by the USB device, the bus interface 207 sends an interrupt signal 318, via the interrupt connection 210 to the processor 206 to inform the processor 206 an error occurred. This completes the IN transfer 306.

In an IN transfer 307, following the IN transfer 304 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the IN transfer by issuing an IN token 319 on the bus. The addressed USB device responds with a NAK handshake 320, indicating it has no data available. This completes the IN transfer 307.

In an IN transfer 308, following the IN transfer 304 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the IN transfer by issuing an IN token 321 on the bus. The addressed USB device responds with a STALL handshake 322, indicating it cannot complete the command. Upon receiving the STALL handshake 322, the bus interface 207 sends an interrupt signal 323 via the interrupt connection 210 to the processor 206. This completes the IN transfer 308.

It will be clear that the IN transfers 304, 305, 306, 307 and 308 are just illustrative examples to clarify the functioning of the bus system and that the functioning of the bus system is limited neither to the IN transfers shown nor to the order in which the IN transfers are shown.

Fig. 3B is a message diagram showing transactions for data transmission. The message diagram illustrates the operation of the system of Fig. 1 for interrupt transactions consisting of OUT transfers. Going top-down the time increases. Information exchange between the processor 206 and the bus interface 207 is symbolized by arrows between a first line 301 and a second line 302. Information exchange between the bus interface and the second station 203, 204 is symbolized by arrows between the second line 302 and a third line 303.

The processor 206 needs to transfer data to one of the second stations 203, 204, a USB device. Thus, the bus interface 207 should repeatedly issue a request to receive

data to the bus connections 205. To reduce the communication overhead for the processor 206 it should be sufficient to have the processor 206 provide the request properties only once. If the addressed USB device accepts data, the bus interface 207 should interrupt the processor via the interrupt connection and inform the processor. In case an error occurs the processor
5 206 should also be notified. In these situations the bus interface 207 should apply an interrupt to the processor 206 too. In other situations no action is required from the processor 206, so that the bus interface 207 should not interrupt the processor 206.

In a OUT transfer 324 the processor in a message 329 sends the request properties to the bus interface 207, where the request properties are stored in the buffer.

10 Based on the request properties within the message 329, which are retrieved from the buffer, the bus interface 207 initiates the OUT transfer by issuing an OUT token 330 on the bus. Subsequently, the bus interface issues a Data packet 331. The second station, the addressed USB device, responds by sending an ACK handshake 332. Upon reception of the ACK handshake 332, the bus interface 207 sends an interrupt signal 333, via the interrupt
15 connection 210, to the processor 206 to inform the processor 206 that the data is transferred successfully. In a different arrangement the bus interface 207 does not send the interrupt signal 332 to the processor 206, since the processor 206 will assume that a successful transmission of the data has occurred if it is not interrupted.

In an OUT transfer 325, following the OUT transfer 324 with identical request
20 properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the OUT transfer by issuing an OUT token 334 on the bus. Subsequently, the bus interface issues a Data packet 335. The addressed USB device responds by issuing a NAK handshake 336 to indicate it is not ready to process the transferred data. This completes the OUT
25 transfer 325.

In an OUT transfer 326, following the OUT transfer 324 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the OUT transfer by issuing an OUT token 337 on the bus. Subsequently, the bus interface
30 issues a Data packet 338. In response the USB device sends a STALL handshake 339 to indicate it cannot process the OUT transfer. In receipt of the STALL handshake the bus interface sends an interrupt signal 340, via the interrupt connection, to the processor 206 to inform the processor on the status of the addressed USB device. This completes the OUT transfer 326.

In an OUT transfer 327, following the IN transfer 324 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the OUT transfer by issuing an OUT token 341 on the bus. Subsequently, the bus interface
5 issues a Data packet 342. Due to an error on the bus, the Data packet 342 is received corrupted by the addressed USB device. For this reason the addressed USB device does not respond. As no response follows by the USB device, the bus interface 207 sends an interrupt signal 343, via the interrupt connection 210, to the processor 206 so as to notify the processor. This completes the OUT transfer 327.

10 In an OUT transfer 328, following the OUT transfer 324 with identical request properties, the bus interface 207 again retrieves the request properties from the buffer 208, while no new information is required from the processor 206. The bus interface 207 initiates the OUT transfer by issuing an OUT token 344 on the bus. Due to the occurrence of an error condition the bus interface does not start or complete an already started issuing of a
15 subsequent Data packet. For this reason the bus interface 207 sends an interrupt signal 345 via the interrupt connection 210, to the processor 206 so as to notify the processor. This completes the OUT transfer 328.

It will be clear that the OUT transfers 324, 325, 326, 327 and 328 are just illustrative examples to clarify the functioning of the bus system and that the functioning of
20 the bus system is limited neither to the OUT transfers shown nor to the order in which the OUT transfers are shown.

Fig. 4 is a block diagram showing a bus interface according to the invention. The block diagram shown in the figure is a block diagram of a bus interface integrated circuit. The bus interface integrated circuit 401 has a dual role. It can be both part of a bus
25 station acting as a USB device and of a bus station acting as a USB host. This is especially useful in a USB On-The-Go (OTG) apparatus. Such a USB OTG apparatus could be, for example, a digital still camera, a digital video camera, a personal digital assistant, or an oscilloscope, which in a certain mode of operation is connected, via USB, for example, to a mass storage apparatus (e.g. a hard disk drive, optical recorder, or solid state recorder), a
30 printer, or a cellular phone. In this mode of operation the USB OTG apparatus should act as a USB host. In another mode of operation the USB OTG is connected, via USB, for example, to a personal computer. In this mode of operation the USB OTG apparatus should act as a USB device. The mode in which the bus interface integrated circuit of the USB OTG

apparatus operates could be software determined and controlled by a processor within the USB OTG apparatus.

In case the USB OTG apparatus is portable it could be advantageous to include a low power consumption mode available in which the bus interface integrated circuits shuts down certain functional blocks or operates at a lower clock speed in time periods with, for example low bus traffic or no bus traffic at all. Such a low power consumption mode could be enhanced by including a so-called remote wake-up feature, controlled by the processor of the USB OTG apparatus, to bring back to operation certain or all functional blocks or to increase the clock speed of the bus interface integrated circuit. The remote wake-up feature could be activated, for example, if the processor anticipates increased bus traffic.

The bus interface integrated circuit 401 comprises a connection 402 with an processor, a connection 403 for connection to a USB in case the bus interface integrated circuit 401 operates as a USB host, and a connection 404 for connection to a USB in case the bus interface integrated circuit 401 operates as a USB device. Note that in a practical implementation connections 403 and 404 could share the same I/O terminals, while an internal multiplexer takes care of the internal routing of the bus signals. Such an implementation could be advantageous in case it is guaranteed that connections 403 are not used simultaneously.

The bus interface integrated circuit 401 also comprises a host-device multiplexer 405, a timing block 406, a host interface 407, a device interface 408, a host controller 409, a device controller 410, a host buffer 411, a device buffer 412, a host transceiver 413, and a device transceiver 414.

The host-device multiplexer 405 is connected to the connection 402 with the processor. Preferably under the control of the processor it determines the internal routing of the communication of the bus interface integrated circuit 401 with the processor. In case the bus interface integrated circuit 401 operates as a USB host, the host-device multiplexer 405 links the host controller 409, via the host interface 407, to the processor. In case the bus interface integrated circuit 401 operates as a USB device, the host-device multiplexer links the device controller 410, via the device interface 408, to the processor. The host controller 409 has access to the bus connection 403 via the host transceiver 413. The device controller 410 has access to the bus connection 404 via the device transceiver 414.

The host buffer 411 is accessible by both the host controller 409 and, via the host-device multiplexer 405, by the processor. It can be used, for example, for storing request

properties issued by the processor and for storing results of the communication between the host controller 411 and a USB device.

The device buffer 412 is accessible by both the device controller 410 and via the host-device multiplexer 405, by the processor. It can be used, for example, for buffering
5 data generated by the processor, so as to be sent to a USB host.

To support co-operation between the bus interface integrated circuit and a wide range of processors, the host buffer 411 and the device buffer 412 are preferably accessible by the processor in both a direct memory access (DMA) mode and a parallel input-output mode.

10 It is advantageous to support several ways of communication between the processor and the buffers within the bus interface integrated circuit 401. This results in a wider choice of processors. For example, a more sophisticated computing system will support DMA operation. If the bus interface also supports DMA operation, the communication overhead of the processor is further reduced. On the other hand, in a less
15 sophisticated bus station the processor handles all communication without the support of a DMA controller. In such a situation the bus interface needs to support the data exchange between the bus interface and the processor in a parallel input-output mode.

Although the embodiment discussed above relates to a bus interface integrated circuit, it will be clear to a person skilled in the art that different blocks or groups of blocks in
20 the diagram could be embodied in different integrated circuits and discrete components. Thus alternative embodiments relating to the bus interface described by the block diagram could comprise a plurality of integrated circuits and discrete components performing the functions of the various blocks.

Depending on the requirements of the bus system it is advantageous to realize
25 the bus interface as an integrated circuit. The functional integration in an integrated circuit enables a reduction in component count and area minimization and hence will result in a reduction in physical size of a bus station comprising the bus interface. Furthermore, realizing the bus interface as an integrated circuit will allow for a reduction in power consumption.

30 Depending on the requirements of the bus system, it is advantageous to realize the bus interface with a plurality of integrated circuits and discrete electronic components, utilizing as much as possible standard building blocks. This reduces the cost price of the bus interface.

Fig. 5 is schematic overview of a buffer 501 organized in a first-in, first-out (FIFO) manner. This in an efficient manner to organize the host buffer 411. The top of the FIFO buffer is indicated by an arrow 502. The bottom of the FIFO buffer is indicated by an arrow 503. The buffer stores request properties of several requests 504, 505, 506, for example, for initiating USB transactions or more in particular USB interrupt transactions. In the overview the request properties first issued are indicated by 504, the request properties issued second are indicated by 505, the request properties issued last are indicated by 506. The area 507 indicates the remaining request properties, while the area 508 indicates the remaining space in the FIFO buffer not being used. Request properties are divided in two parts: a transfer descriptor or header 509 and, if applicable, payload data 510. The transfer descriptor 509 describes the characteristics of the transaction to be handled by the host controller 409. The payload data 510 is the data that should be included in the Data packet of the transaction to be handled by the host controller. The transfer descriptor is also used by the host controller 409 to store the results of transactions. Therefore, both the processor and the host controller 409 have access to the FIFO buffer 501.

It is advantageous to arrange the buffer in a first-in first-out mode. In this way it is relatively simple to access the data stored in the buffer for both the processor as the controller.

It is advantageous to use the buffer to store both request properties and the results of the communication with the second station. By giving the processor read access to the buffer, or at least designated sections of the buffer, an efficient way of communication is established, since the same buffer is used to store the request properties and the results of the requests. Thus, the processor can access the buffer in an identical way as when it writes the request properties, only the type of access has to changed from a write mode to a read mode.

It is advantageous to have the bus interface determine the exact polling interval based on information provided by the processor. Limiting the number of possible polling rates simplifies the task of the bus interface in handling different requests with different polling rates. For example, if the processor may choose any polling rate between 1, 2, 3, and 128 milliseconds, the bus interface maps the chosen polling rate to one of the following actual polling rates: 1, 2, 4, 8, 16, 32, 64, or 128 milliseconds, thus reducing the potential number of different polling rates to handle from 128 to 8. Preferably, the bus interface maps the polling rate chosen by the processor on the actual polling rate that is closest to the chosen polling rate.

The table below shows a format of the transfer descriptor. The format shown consists of 8 bytes, indicated as Byte 0, Byte 1, ..., Byte 7. Every byte consists of 8 bits, enumerated as 0, 1, ..., 7.

-	7	6	5	4	3	2	1	0
Byte 0	Actual bytes (7:0)							
Byte 1	Completion code				Active	Toggle	Actual bytes (9:8)	
Byte 2	MaxPktSize (7:0)							
Byte 3	Endpoint number (3:0)				Last	Speed	MaxPktSize (9:8)	
Byte 4	Total bytes (7:0)							
Byte 5	0	0	Res	Res	Dir. token (1:0)		Total bytes (9:8)	
Byte 6	0	Device address						
Byte 7	Interrupt polling interval (4:0)							

The table below clarifies the meaning of the parameters in the table above.

Name	Length (bits)	Description
ActualBytes	10	Contains the number of bytes that were transferred
Completion code	4	Indicates how the last transaction for the involved transfer descriptor was completed, including occurrence of errors
Active	1	Indicates whether or not transactions should be executed
Toggle	1	Used to generate or compare the data PID value. It is updated after each successful transmission or reception of a Data packet
MaxPktSize	10	Indicates the maximum number of bytes that can be sent to or received from the involved endpoint in a single Data packet
Res	1	not used
Speed	1	Indicates the speed of the involved endpoint
TotalBytes	10	Indicates the total number of bytes to be transferred
Dir. token	2	Indicates if the involved transfer descriptor involves IN transfers or OUT transfers
Device address	7	Indicates the address of the USB device that should be addressed
Interrupt polling interval	4	Indicates the polling interval of the interrupt transfers

CLAIMS:

1. A bus system comprising a first station (202) and a second station (203, 204) coupled by a bus for transferring data and control signals, said bus being conceived to operate according to a protocol in which said first station (202) repeatedly sends requests to said second station (203, 204), characterized in that said first station (202) comprises a processor (206), a bus interface (207), and a buffer (208) coupled to said processor (206) and said bus interface (207), where
- 5 - said processor (206) is operable to generate request properties for said requests and for handling said data items;
- said buffer (208) is operable to store said request properties;
- 10 - said bus interface (207) is operable to generate said requests from said stored request properties and to repeatedly send said requests.
2. A bus system as claimed in claim 1, characterized in that said request properties comprise a polling rate of said requests.
- 15 3. A bus system as claimed in claim 2, characterized in that said request properties also comprise a payload of said requests.
4. A bus system as claimed in claim 1, characterized in that said second station (203, 204) responds to said requests, that said processor (206) is an interruptible processor and that said bus interface (207) is operable for interrupting said processor (206) upon reception of selected responses of said second station (203, 204).
- 20 5. A bus system as claimed in claim 1, characterized in that said bus system is a USB system.
- 25 6. A bus interface comprising a connection for a bus (403) and an input for receiving request properties from a processor, characterized in that said bus interface comprises a buffer (411) operable to store said request properties and a controller (409)

operable to generate requests out of the stored request properties and to repeatedly send said requests to the connection (403).

5 7. A bus interface as claimed in claim 6, characterized in that said request properties comprise a polling rate of said requests.

8. A bus interface as claimed in claim 7, characterized in that said request properties also comprise a payload of said requests.

10 9. A bus interface as claimed in claim 6, characterized in that said bus interface also comprises an interrupt output for sending an interrupt signal to an interruptible processor, and that said controller (409) is operable to receive responses from said connection and to interrupt said processor upon reception of selected responses by sending an interrupt signal to the interrupt output.

15

10. A bus interface as claimed in claim 6, characterized in that said bus interface is operable to operate as a USB interface.

1/6

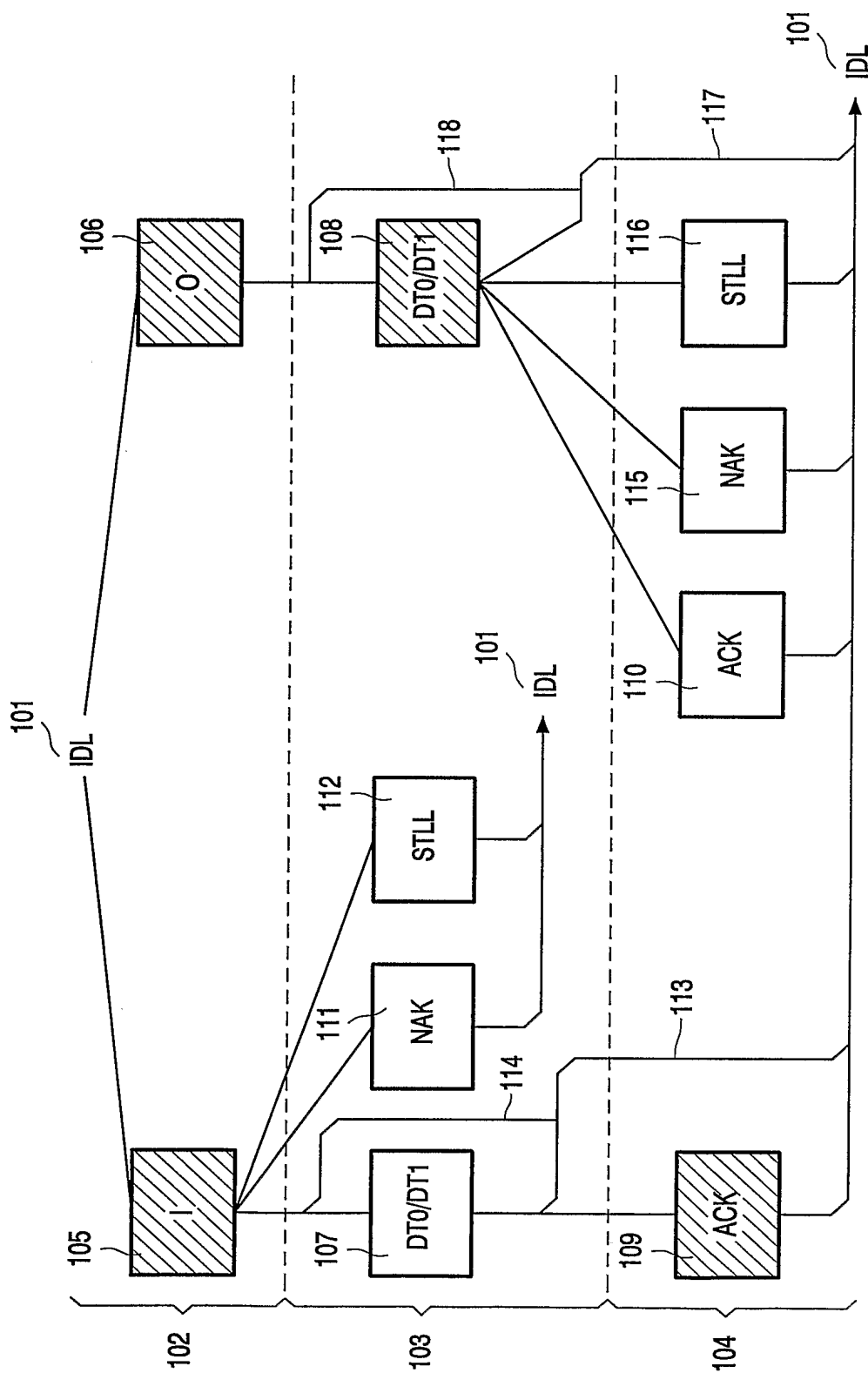


FIG. 1

2/6

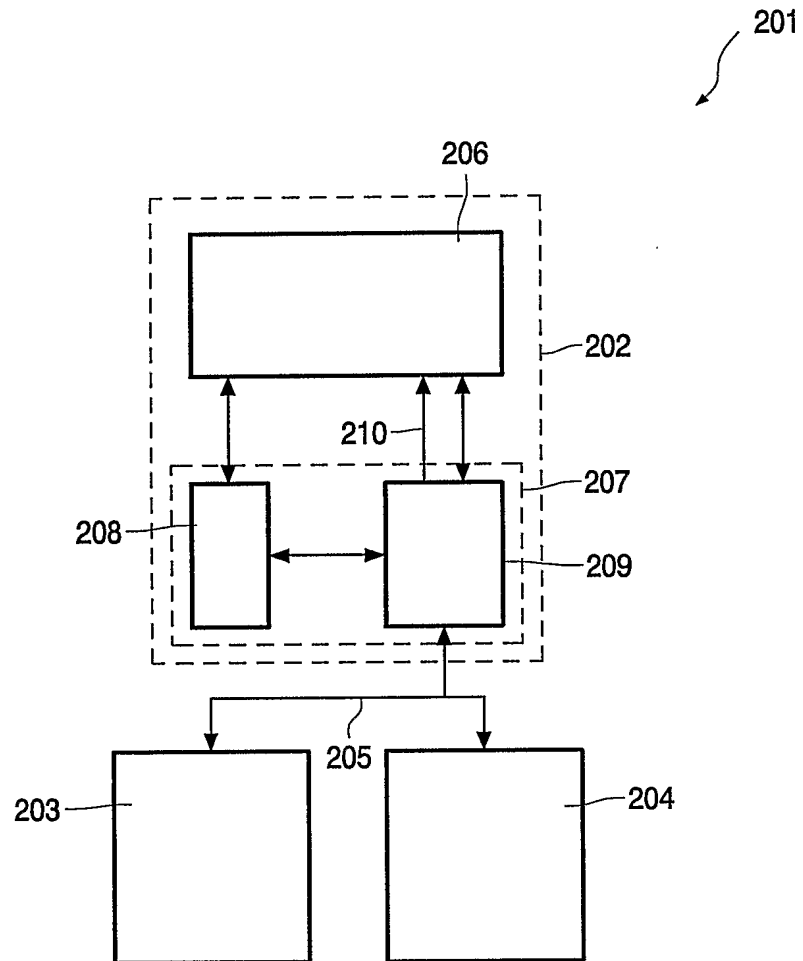


FIG. 2

3/6

IN

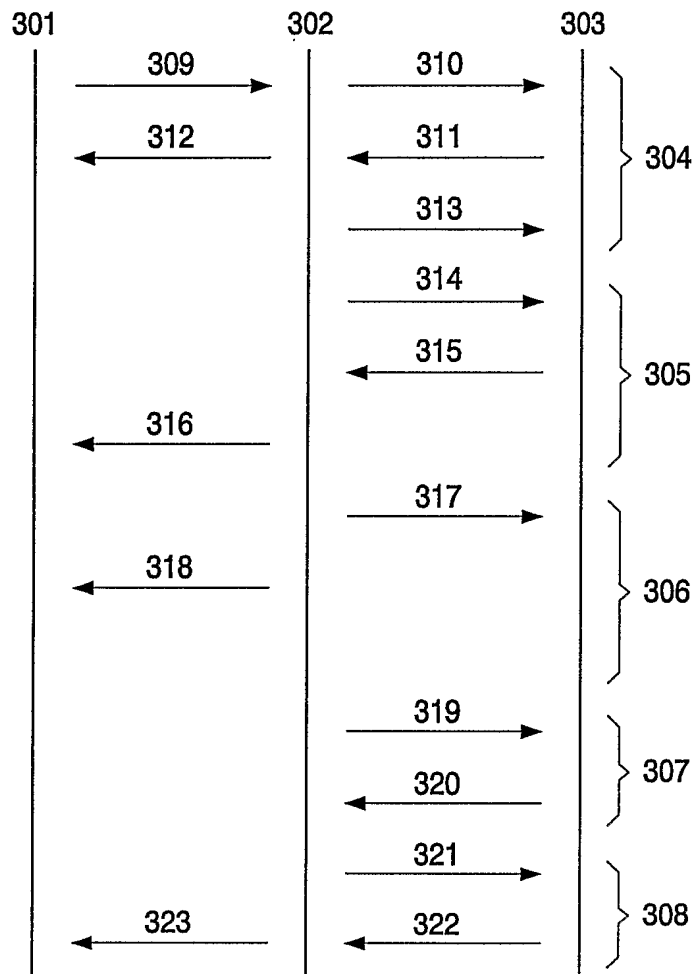


FIG. 3A

4/6

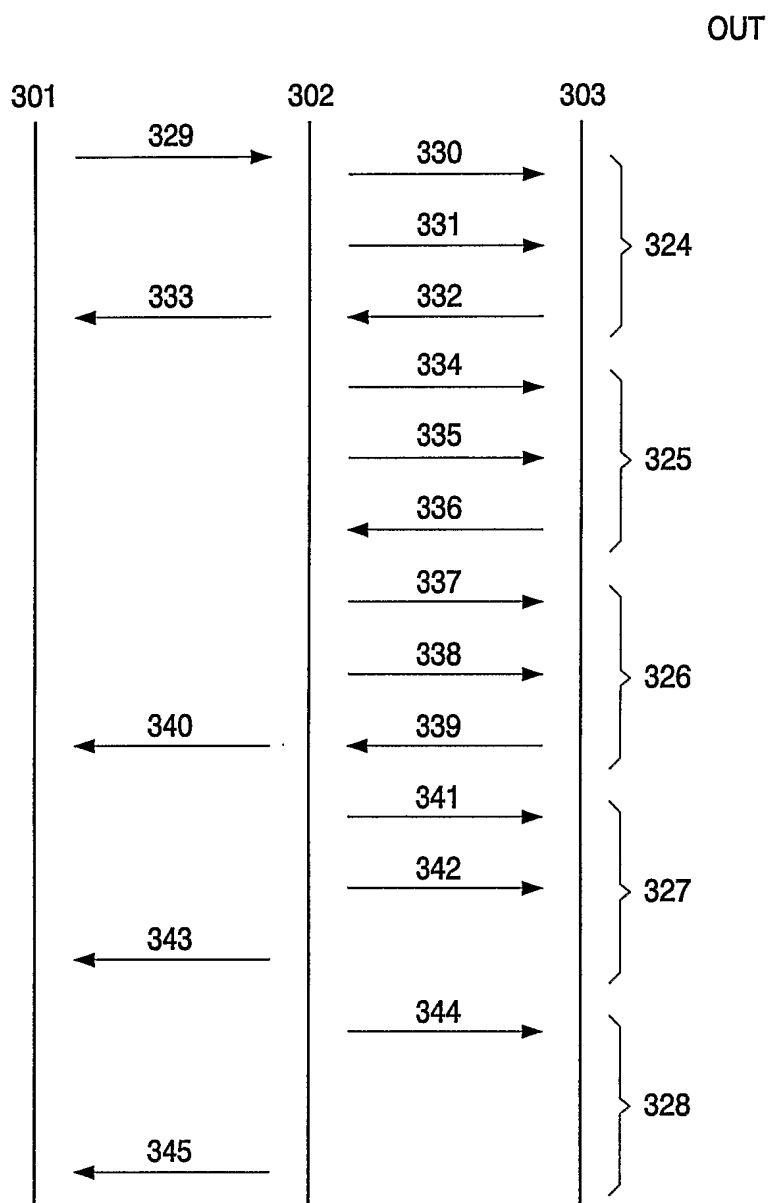


FIG. 3B

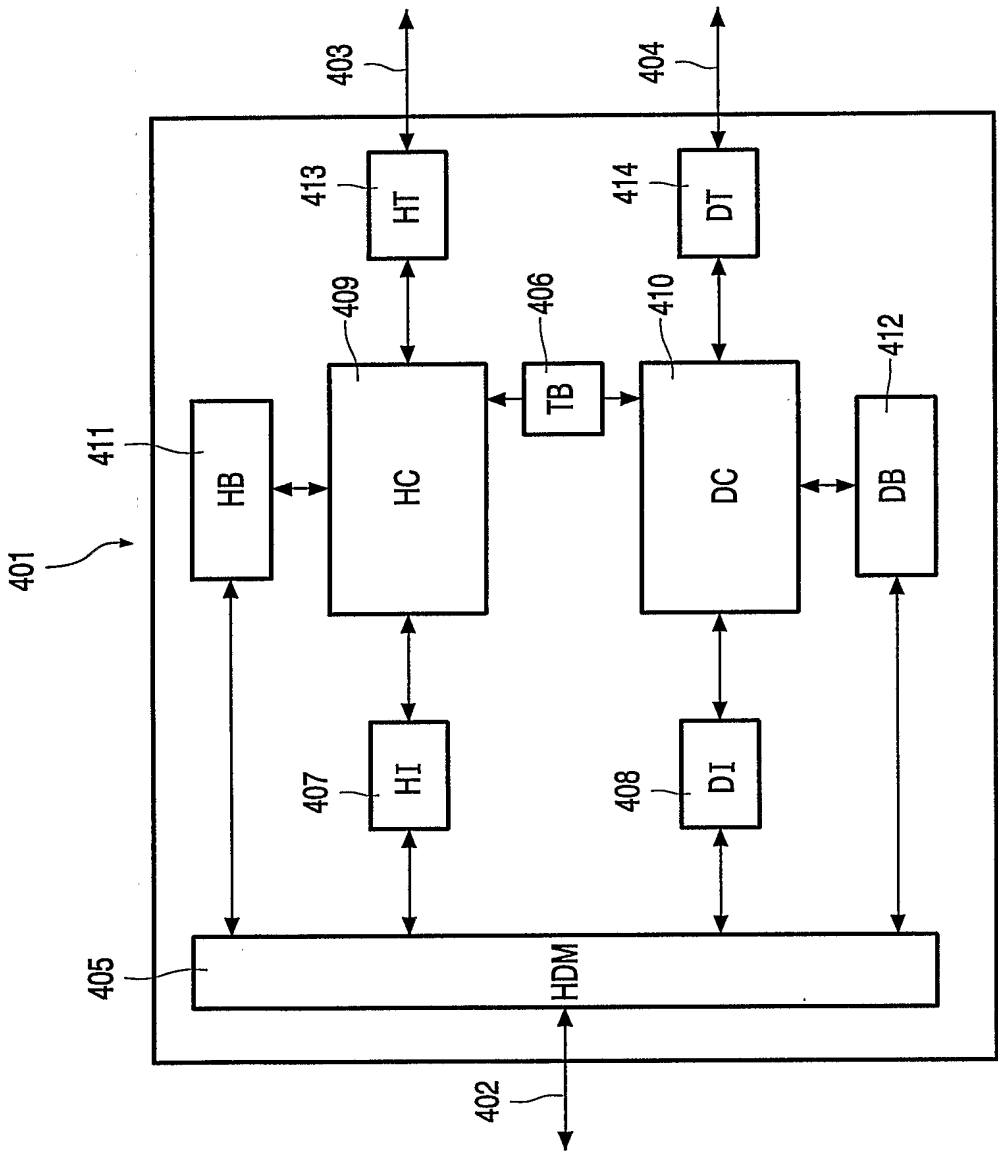


FIG. 4

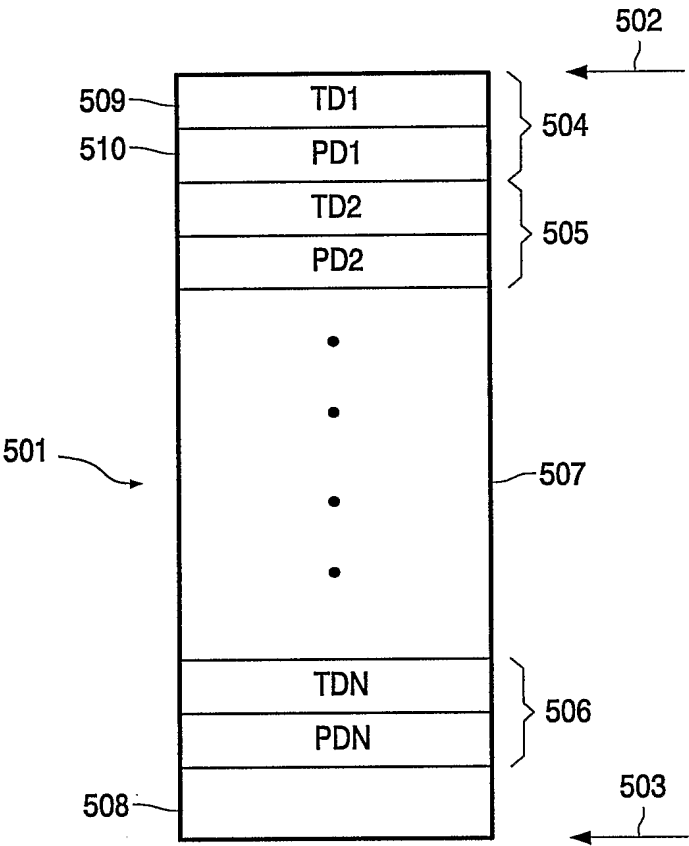


FIG. 5

INTERNATIONAL SEARCH REPORT

International Application No

PCT/IB 02/03812

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F13/24 G06F13/42

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, IBM-TDB, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6 145 045 A (BRIEF DAVID ET AL) 7 November 2000 (2000-11-07) column 2, line 9 -column 3, line 48; figure 1	1,4-6,9, 10
Y	ANONYMOUS: "Universal Host Controller Interface (UHCI) Design Guide" 'Online! March 1996 (1996-03) , INTEL CORP. XP002225237 Retrieved from the Internet: <URL: http://developer.intel.com/design/USB/UHCI 11D.htm> 'retrieved on 2002-12-17! page 5 -page 6	1,4-6,9, 10

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

- *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- * & * document member of the same patent family

Date of the actual completion of the international search

17 December 2002

Date of mailing of the international search report

13/01/2003

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer

Henneman, P

INTERNATIONAL SEARCH REPORT

International Application No

PCT/IB 02/03812

Patent document cited in search report	Publication date	Patent family member(s)	Publication date	
US 6145045	A	07-11-2000	DE 19900331 A1	08-07-1999
		US 6173355 B1		09-01-2001